Holistic discovery of decision models from process execution data

Holistic Discovery of Decision Models from Process Execution Data

Johannes De Smedt

University of Edinburgh Business School Management Science and Business Economics Group, 29 Buccleuch Place, EH8 9JS, Edinburgh, UK

Faruk Hasić, Seppe K.L.M vanden Broucke, Jan Vanthienen

KU Leuven Faculty of Economics and Business Department of Decision Sciences and Information Management, Naamsestraat 69, 3000 Leuven, Belgium

Abstract

The analysis of business processes is a multifaceted problem that is comprised of analysing both activities' workflow, as well as the decisions that are made throughout that workflow. In process mining, the automated discovery of process models from event data, a strong emphasis can be found towards discovering this workflow, as well as how data influences that workflow, i.e., decision point analysis. Nonetheless, the data that is pertaining to the activities in the workflow does not necessarily correlate with the control flow. Decisions that influence variables that are used by activities can also impact other variables used later in the workflow without interfering with the order in which activities are executed. Discovering this has not been addressed in literature, as current decision mining techniques still rely on control flow.

To address this, Process Mining Integrating Decisions (P-MInD) is proposed. It relies on uncovering the influence of activities on their variables and connects them by making use of sequence dependencies present in the data. Furthermore, it allows to find autocorrelations, as well to incorporate case variables. This allows to establish a holistic image of the decision layer, captured with Decision

*Corresponding author
Email address: Johannes.DeSmedt@ed.ac.uk (Johannes De Smedt)

Model and Notation (DMN), that is consistent with the discovered control flow. Furthermore, decision model conformance checking, i.e., the matching of event logs with holistic models, is proposed to offer a way to verify whether the models are corresponding with the behaviour that is present in the current system. P-MInD is implemented and used on real-life data to verify its effectiveness.
Keywords: Decision mining, Process mining, Decision Model and Notation, DMN.

---

1. Introduction

The contemporary surge of data collection within various systems and in a plethora of formats has urged data scientists to establish data mining techniques that are tailored towards a variety of settings. One such setting, which has a strong presence within organizations, are processes. Processes are stored on a case basis, i.e., they report on the execution of various activities (events) pertaining to a certain entity, e.g., a patient, an insurance claim, and so on. These data, sequences of events stored in a case-based fashion, require appropriate techniques to retrieve insights adequately. Process mining is a popular new topic [1] which is regarded as a novel approach established to analyse the execution of processes within an information system. Applications for process mining and subsequent analysis can be found in a number of areas, such as back-end operations (call center routing), hospitals (care paths), manufacturing (production), and so on.

The research area, however, has long been dominated by a strong emphasis on the control flow, i.e., retrieving models to capture the execution sequences by means of concurrent models such as Petri nets [2] or Business Process Model and Notation (BPMN) models [3]. While these outcomes are suitable for analysing deviating process executions and bottlenecks, throughput time, and other scheduling-based questions, the data that is used and stored throughout still hides considerable information that is left untouched. One of the major attempts to uncover the insights that are recorded next to behavioural data in

information systems, was the use of decision mining in processes as introduced in [4]. By using variables tied to activity execution, decision mining is able to build predictive models that explain why certain paths in a process are followed at fixed locations in the workflow that allow for exclusive choice (so-called XOR-gates or -splits). Essentially, this approach is still very much control flow-driven and can be considered as decision point analysis. Many other techniques have been proposed to improve upon the first attempts in order to solve problems regarding the analysis of loops and other routing constructs such as invisible activities. Nevertheless, establishing the discovery of how the control flow is driving the variables, and how the variables are interrelated throughout the process, can only be captured when dedicated models are introduced.

Recently, the uptake of research on decision modelling in a process context has led to initiatives such as Decision Model and Notation (DMN) [5], a counterpart of BPMN to capture the relation between activities, variables, and how they establish decisions in a workflow. As such, a process model's decision layer is offloaded from including intricate variable and relational information, whilst increasing the expressiveness to include decision logic in a process and retaining a separation of concerns.

To complement the insights from decision mining, new approaches have introduced the discovery of DMN models from process data [6, 7]. Nevertheless, the focus remains with explaining the control flow, or at least with models that are incorporating control flow constructs. In this paper, Process Mining Integrating Decisions (P-MInD) is introduced. P-MInD mines the relationship between activity variables and does so in a control flow-agnostic, but control flow-compliant fashion, i.e., behavioural information is used to ensure decisions orders are corresponding with the behaviour seen in the event log. This ensures P-MInD adheres to the separation of concerns, while obtaining insights that previously remained hidden.

P-MInD contributes to the area of decision model mining in processes literature by introducing an approach to mine for decisions independent of, but consistent with the control flow which can incorporate case attribute, as well

as discover autocorrelations of variables that appear in activities. Furthermore, by using the concept of shifts, it can reveal how variables are sometimes influenced, and sometimes not influenced when used by activities that occur in loops. All these elements constitute a stronger, more in-depth view of the decisions that are made in a process, which we refer to as holistic. Finally, this paper also introduces a decision model-driven conformance checking approach, which can validate all the separate decision models that comprise the holistic decision model.

P-MInD utilizes the concept of Variable-Activity Pairs (VAPs) to constitute decisions in event logs, i.e., the combination of an activity influencing a variable in the process. These VAPs are retrieved from the process execution log (hereafter called event log), taking into account the occurrence of the location within the execution, a particular shift caused by a VAP, to pinpoint which VAPs are related to which VAPs within the trace. This allows to construct dependencies taking into account the sequence which is relatable to the control flow model (i.e. process model) and use the reoccurrence of an VAP to mine autocorrelations. To establish the relation between an activity and its variables, correlation or the structure of predictive models is used. Depending on the number of occurrences and the execution order of the activities, the influence over the variables is different. Therefore, the traces that exhibit a similar VAP-pattern are clustered to find models to ensure consistency between control flow and decision model(s). This approach is desirable within event logs, as the variation in execution paths is typically correlated with the case's characteristics, e.g., extraordinary patients exhibiting an unusual ailment might be facing a care path that deviates from other patients. Hence, the VAPs contribute significantly to the holistic nature of the discovered decisions models, as they allow for the discovery of autocorrelation and loops through splitting up activity occurrences. The findings are implemented in the Process Mining (ProM) framework and is available in the latest release.

This paper extends the findings of [8] where the initial version of P-MInD and the concept of a holistic process model was introduced and strengthens it

in various ways. First of all, it introduces variable-attribute pairs to capture relations, which makes it possible to capture autocorrelations and also explains which shift, i.e., which appearance of an activity influencing a variable, is responsible for the relationship between activities and variables. Secondly, it introduces a conformance checking approach, which also applies to the models introduced in [8]. Finally, this paper also reports on the incorporation of case attributes into decision models.

The paper is structured as follows. Section 2 illustrates why integrated, holistic decision discovery is necessary and how it relates to other techniques with a running example, completed with other related work and the contributions of this paper. The main concepts for decision mining are discussed in Section 3 and applied in Sections 4 and 5 that introduce the discovery algorithm and conformance checking approach respectively. Section 6 introduces the implementation which is used for empirical evaluation subsequently. Section 7 wraps up the paper with a discussion of results and future work.

2. Motivation, related work, and contributions

In this section, an example process containing various types of decisions is illustrated. Next, the presence of decision points, as well as decision models is discussed. For a full survey on the various types of decisions in processes and how to mine them, we refer to [9]. Furthermore, this section addresses related work in the fields of decision modelling and data-centric process approaches. Finally, the section concludes by outlining the main contributions of this paper.

2.1. Illustrative example

Consider the example in Figure 1, representing a procurement process in Colored Petri nets (CPN) [10]. The process starts off with the receipt of an order. At that moment, the value of the order (v), and type (t) are set and passed on through the process. Once the purchase is confirmed, the shipping method is determined. Afterwards, the shipping price is determined as well

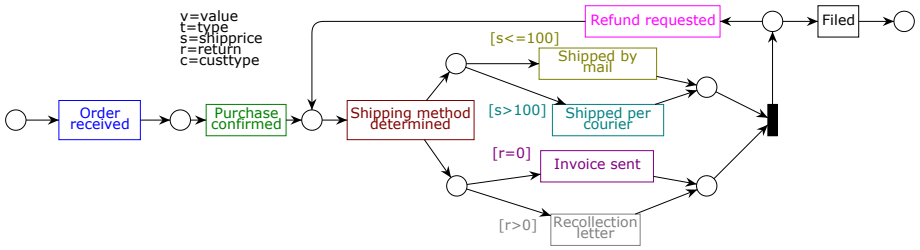

Figure 1: Running example of a procurement process in which a customer can return a product captured in a colored Petri net. Note that all the arcs should carry all the variables, but are omitted for clarity.

(s). Note that this entails two different decisions. The former is a control flow decision, i.e., the routing of what activity to execute is decided based on the shipping price. To obtain the latter, a decision is made based on the type (t) and value (v) to find a value for s. Afterwards, both s and r can influence the routing, but the decision of s is independent of the control flow. After shipping, the customer can determine whether a refund is necessary. Possibly, a new product is shipped, or a recollection letter. By setting the refund variable r, the control flow is influenced. Finally, once all customer-based activities are performed, the order is filed and classified. A decision is made on what type of customer the process was dealing with. The outcome is stored in variable c. Again, this variable is set based on the value of previously set variables, such as s, t, r, and v.

In Table 1, a potential event log of two cases is depicted which illustrates these decisions. In this case n refers to the case id to identify what customer is dealt with. First, upon entering the system, *Order received* (OR) sets t and v. These values will be used later on to determine the shipping value s in *Shipping method determined*. In both cases, an invoice is sent, and the product is shipped by mail. In case 1, however, a refund is requested. Again, the shipping method is determined, and the customer does not have access to corporate discounts resulting in a higher shipping cost s. Since it pertains to a refund, a recollection letter is sent. Finally, in both cases the customer lifetime value is calculated by

activity *Filed*.

a	n	v	t	s	r	c
OR	1	10	1	0	0	0
PC	1	10	1	0	0	0
SMD	1	10	1	6	0	0
SM	1	10	1	6	0	0
IS	1	10	1	6	0	0
RR	1	17	2	6	1	0
SMD	1	17	2	8	1	0
RL	1	17	2	8	1	0
SC	1	17	2	8	1	0
F	1	17	2	8	1	89

a	n	v	t	s	r	c
OR	2	200	3	0	0	0
PC	2	200	3	0	0	0
SMD	2	20	3	15	0	0
SC	2	20	3	15	0	0
IS	2	20	3	15	0	0
F	2	20	3	15	0	514

Table 1: Event log generated by executing the process model of Figure 1.

2.2. Decision point analysis

Decision point analysis focuses on the XOR-splits that can be found within a workflow (typically Petri nets), typically mined from an event log [4]. These splits resemble decisions towards executing a particular activity within the process. By using the data that is available next to the activities, i.e., the event data, as independent variables within predictive models that use the activities present after the decision point as a discrete dependent variable. Many issues remain, most notably finding the correct data that is of relevance to the decision point for which a model is produced. Consider the example, the places after activity *Shipping method determined* are decision points (depending on the definition used, one can claim the activity itself is the decision point) in which it is decided what type of shipping needs to be used and whether an invoice or recollection letter needs to be sent. The rules that can be retrieved for this, are included in the guards on the activities. In trace 1, however, this decision is made twice. Hence, it remains to be seen what data is to be used. A simple heuristic is to use the latest data only, or an alternative is the usage of all previous data up until that point. Hence, loops, or repetition of behaviour, is hard to interpret in decision points [4]. E.g., in [11], the latest data is used. Next, the presence of hidden activities, i.e., activities without a label that are included for routing purposes, make it unclear what activities that are connected to them

further in the execution as their number can grow quickly. Besides, loops come into play again, as activities decide over themselves. Finally, overlapping rules, i.e., rules that point to multiple activities after the decision point are a problem as well. A solution for overlapping rules was found in [12].

Many extensions and improvements to decision point analysis were proposed since, e.g., by using improved inference techniques [13] and alignments [14] and finding overlapping rules [12]. An overview of many of these works can be found in [15]. The learning of Markov models to capture decision point outcomes is similar to many of the Petri net approaches as well [16].

2.3. Decision model mining

Decision models focus not only on the control flow aspect, but rather on all decisions that are made within the workflow. The DMN standard visualizes decision models as a hierarchical, acyclic model that connects decisions with their input variables, the knowledge required to make a decision, and the other decisions it provides a subdecision for. To be able to connect the decision models with process models, a common ground is found in the activities in the process model. Indeed, activities make decisions within processes. Decision point analysis takes the activities placed before decision points (places within a Petri net or gateways in BPMN) as a decision. This can be linked to DMN models, as in [7]. However, this still only reflects on single decisions and separate DMN models. A decision model encompasses the full decision lifecycle of a process, including multiple decisions and their relationships. A more holistic approach would be to connect the variables of various decision points, and how they affect each other. Such an approach was provided by [6], work based on [7] and [17], which bridges the gap between various decision points and predefined relationships. In the example, however, no multiple decision points are present, except for the repeating one at *Shipping method determined*. The technique, however, is not capable of dealing with loops. It would be capable of finding relationships between attributes, e.g., if the shipping price is dependent on the type and value of a product. Nonetheless, the decision on the shipping price

would still need to be linked to an ad-hoc generated decision, or a decision point, as all variables/attributes are compared with each other globally. Hence, the control flow is still enclosed within the decision models, something that does not correspond with the setup of the DMN standard's view on integration between decisions and processes.

2.4. Data-centric methods, decision modelling, and related approaches

Bayesian approaches for decision modelling in process execution (analysis) exist as well, most notably [18] uses Bayesian inference networks to predict decision outcomes, and [19] construct belief networks for event logs. The former discusses how to represent and predict decision model outcomes from process executions, but does not relate to activities directly. The latter are similar to P-MInD in that they establish networks of activities and variables, but the problem of loops is not addressed. Furthermore, the technique only uses parametric models and has the main purpose of still discovering control flow models, rather than decision models.

Conformance checking of discovered decisions has not necessarily been addressed directly, but data-aware approaches such as [20] and [21] can be used to analyse either data-enriched, or decision point-based process models.

Contrary to focusing on the control flow, other works exist that rather start from the data perspective while either incorporating control flow for clarification, or by structuring the results. In [22] a general framework for correlating business activity variables and process variables is proposed, and in [23], the resource perspective is mixed with the control flow for recommendations of future executions.

In the more general research area of decision modelling many works are focusing on DMN and other decision modelling frameworks such as Product Data Models [24, 25], as well as the work on multi-perspective object-oriented process models such as Guard-Stage-Milestone [26] and its discovery [27] and Object-Centric Behavioural Constraint Models [28]. In [9], a framework to position efforts on decision modelling and mining in processes is proposed. Many works

focus on capturing business rules in decision tables [29, 30] and the separation of concerns between process and decision model [31, 32, 33] has drawn an extensive part of recent literature. DMN has also been applied in the context of disaster management [34].

2.5. Contributions

P-MInD [8] rather uses the relationship between activities and variables to establish decisions. A combination of an activity deciding on a particular variable is used as a decision, meaning that activities can appear in multiple decisions. The variables serve both as an input to the decision, as well as the output (the variable that is decided on) and allow for the construction of a decision model. In the example, *Shipping method confirmed* decides on the shipping price (s), and the *Filed* activity decides on the customer type (c). Hence, two decisions are made, which are also connected through s, as information regarding the shipping can influence the type of the customer c. By starting from the activities to build models or establish correlation, not all variables need to be compared or used in different combinations in a possibly very large variable space. Besides, P-MInD takes into account the precedence between the occurrence of variable-activity pairs forming decisions to be consistent with the workflow. In this work, a more rigorous approach is devised to connect decisions by using, instead of precedence, the shifts of a variable. Every change in the value of a variable is recorded and linked to a decision. This renders P-MInD capable of pinpointing what position within a trace the decisions are made, and allows for introducing autocorrelation, i.e., decisions influencing their later instantiations. It also allows to deal with loops and picking up the correct information for different versions of decisions. Finally, the approach naturally finds the decision inputs, i.e., other decisions, that correspond with the traces in which the shifts occur. That way, all models that are built out of the event log are bundled in clusters of traces, effectively bundling the integrated decision and process models in clusters in which the control flow is completely aligned with the data flow (as in the partial ordering of DRDs) along the different shifts.

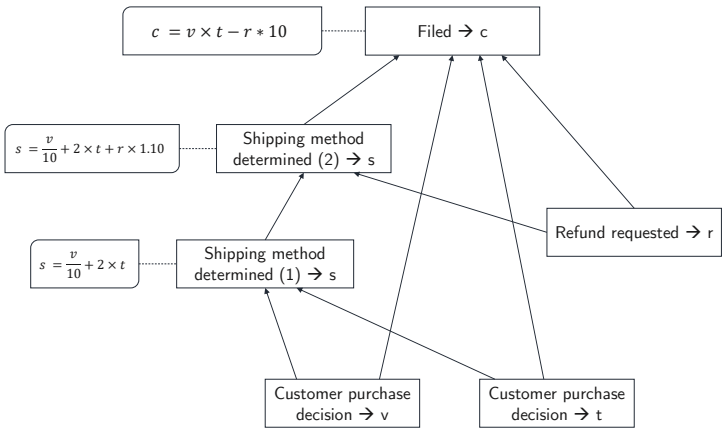

Figure 2: Decision Requirements Graph capturing the decision model present in the Petri net extended with decision logic.

A decision model, more specifically a Decision Requirements Graph (DRG) is visualized in Figure 2 to illustrate what decisions are present in the process.

3. Preliminaries

In this section, the concepts of event logs, activities, and decisions are devised.

3.1. Decisions

Definition 1. *A decision $d \in D_{dm}$ is a tuple (I, O, L), where $I \subseteq ID$ is a set of input symbols, O a set of output symbols and L the decision logic defining the relation between symbols in I and symbols in O with $I \cap O = \emptyset$.*

In case of decision tables, a commonly used reasoning construct in decision models, I and O contain the names of the input and output elements, respectively, and L is the table itself, i.e., the set of decision rules present in the table. As of now, we consider the decision logic L to be a predictive model that links every input to every output. In DMN, decisions and decision logic make up the nodes of a DRG, which are defined as follows.

Definition 2. *A Decision Requirements Graph is a directed acyclic graph* $(D_{dm} \cup ID \cup L_d, E)$*, that connects the decisions, decision logic, and inputs with edges* $E \subseteq D_{dm} \times D_{dm} \cup L \times D_{dm} \cup ID \times D_{dm}$*.*

As illustrated in the example in Section 2, the decisions are represented as squares, where knowledge nodes are illustrated as semi-rounded parallelograms. Inputs are not used in this work, but are typically represented as rounded rectangles in DRGs.

3.2. Event Logs

Process mining and its related techniques employ the notion of event log to define the structure of data suitable for activity- and case-based discovery.

Definition 3. *An event log is a tuple (E, A,* λ*, V, var,* Val*,* $\mathcal{L}$*), where:*

- E *is a set of events.*
- A *is a set of activities (event types).*
- $\lambda : E \to A$ *is a labelling function mapping events to activities.*
- V *is a set of variables.*
- $var : E \to \mathbb{P}(V)$ *is a function mapping events to the subset of variables used in this event.*
- *For each* $v \in V$ *a partial function* $val_v : E \to dom_v$ *mapping events to values in the domain of* v*. We denote the set of these partial functions as* Val*.*
- $\mathcal{L} \subseteq \bigcup_{n \in \mathbb{N}} E^n$ *a non-empty set of event tuples in the log which contains traces* $\sigma \in \mathcal{L}$ *which are sequences of events* $\sigma = \langle e_1, ..., e_{|\sigma|} \rangle$*.*

For the sake of brevity we use $var(a) = \{v \in var(E) | \lambda(E) = a\}$*, i.e.,* $var(a)$ *for* $a \in A$ *denotes all variables appearing with the events with label* a*.* σ_a *is a subsequence of* σ*,* $\sigma_a \subseteq \sigma$ *where* $\sigma_a = \langle e_i \in \sigma \mid \lambda(e_i) = a, a \in A \rangle$*.*

Typically, special variables include the timestamp ($t \in V$) and resource ($res \in V$). The timestamp is denoted as $T(e) = val_t(e)$.

3.3. Business Activities

Decisions do not surface solely as the driver of control flow. Rather, they steer the routing of cases because of decision outcomes that are needed as inputs to other decisions, but also make changes in the data layer of the process without influencing the cases' workflow directly. The latter introduces numerous types of activities that are representatives of the *decision* model in the *process* model:

Definition 4. *The input and output data variables of business activities are defined as follows:*

- $I : A \to \mathbb{P}(V)$*, function mapping a certain set of variables to the input of a certain activity,*
- $O : A \to \mathbb{P}(V)$*, function mapping a certain set of variables to the output of a certain activity.*

This enables the construction of the following activity types:

1. **Operational activities (inputs (optional), no outputs):** do not have any influence on the process' decision dimension and only act as a performer of a specific action that is tied to that specific place in the control flow. They might serve as the end of a decision. They are provided with the decision inputs needed, which are not used further in the process, $A_o = \{a \in A \mid O(a) = \emptyset, \}$.
2. **Administrative activities (no inputs, outputs):** have the purpose to introduce decision inputs into the process, $A_a = \{a \in A \mid I(a) = \emptyset \wedge O(a) \neq \emptyset\}$.
3. **Decision activities (inputs, outputs):** serve a true autonomous decision purpose as they transform decision inputs into a decision outcome, $A_d = \{a \in A \mid I(a) \neq \emptyset \wedge O(a) \neq \emptyset\}$.

It holds that $A_a \cup A_o \cup A_d = A$. Typically, the decision points that are used for decision mining in processes are of the decision activity type, but tailored

towards deciding which activity should be performed next based on the event labels. Note that these are not included in V.

We can now make the connection with decisions and decision models. For discovery purposes, we add the notion of shift and correlation to obtain the final version of decisions discovered from event logs:

Definition 5. *A decision d in a process is a tuple (I_d, a, o_d, L_d) with:*

- $I_d \subseteq \{a_i \in A_a \cup A_d \mid O(a_i) \in I(a)\}$ *the possible inputs,*
- $o_d \in O(a)$ *with $a \in A$, and*
- $L_d : I_d \times o_d \to \mathbb{R}$ *a predictive model assigning a real-valued evaluation metric.*

Many activities within a process can be linked to different decisions. This insight is something that is in contrast with control flow-based decision model approaches as they are tied with activities as decisions. Other decisions serve as inputs as they set variables that serve as an input for a. Hence, inputs are replaced by administrative activities. Note that there is only one output node, contrary to the more general formulation in Definition 1 to conform with the setup of predictive models with a single dependent variable. However, the same activity can have the same inputs for multiple outputs in different decisions.

4. P-MInD

P-MInD follows three main steps which are outlined in detail below. First, the event log is scanned to register all shifts of variable values. Next, these shifts are used to discover the different DRGs. Finally, the models are merged and returned per top-level activity decision. This section is structured accordingly.

4.1. Finding shifts

To capture the dynamics of variables in event logs and the influence of activities over them, the concept of shifts is introduced.

Definition 6. *A shift sequence* $\sigma^{v,a} \subseteq \sigma_a$ *is a subsequence of events of* a *in trace* σ *where* $\sigma^{v,a} = \langle e_i \in \sigma_a \mid val_v(e_i) \neq val_v(e_{i-1}) \rangle$*, i.e., all occurrences of* a *where the previous value of* v *is different from the current value. A shift* $s^{v,a}_{\sigma,i}$ *is the* i*th of those occurrences in* $\sigma^{v,a} = \langle s^{v,a}_{\sigma,1}, ..., s^{v,a}_{\sigma,|\sigma^{v,a}|} \rangle$.

We denote $S^{v,a} = \{s^{v,a}_{\sigma,i} \mid s^{v,a}_{\sigma,i} \in \sigma^{v,a}, \forall \sigma \in \mathcal{L}\}$ all shifts for the variable-activity pair (v, a), and $S^{v,a}_n$ all the nth shifts of (v, a). Separate shift sequences for σ^v can be constructed similarly.

To capture the shifts, Algorithm 1 is operationalised. All parameters are indicated in green. First, the event log is traversed and for every event it is checked whether a shift has occurred (Algorithm 1 lines 3-11). In case the previous value of a variable is different (line 8), this is recorded in the sequence of shifts $\sigma^{v,a}$. All shift sequences are stored for later use and querying according to Definition 6. The first event of a trace is considered responsible to introduce the variables, and hence also captures case attribute values which are considered constant over the trace σ, i.e., $val_v(e_i) = val_v(e_j), \forall e_i \neq e_j, \forall e_i, e_j \in \sigma$.

Next, in lines 12-17, it is determined whether the shift ratio of a VAP is high enough, i.e., whether the activity alters the value of a particular variable enough times to assume there is an influence over the variable. The sensitivity of the algorithm to perform this filtering is set by the `minShift` parameter, which is a percentage. All VAPs that exceed the parameter's value are retained for building predictive models later. Other variables that are not necessarily set, i.e., there are too few shifts but they are set by the activity and contain at least 2 values (meaning they can be discriminative in a predictive model), are kept as possible inputs of the activity.

4.2. Connecting activities

To capture the relationship between activities through their variables for a certain shift, we introduce DMN models for mining as follows:

Definition 7. *A DMN model for mining is a tuple* (D, E, C, L, T) *with:*

– $D \subseteq V \times A \times \mathbb{N}$ *the set of Variable-Attribute-Shift number (VAS) tuples,*

Algorithm 1 Mining decision models from an event log

1: **procedure** MINE_DMN_MOD($\mathcal{L}$) ▷ Input: Log and parameters
2: $DRG \leftarrow \emptyset$ ▷ DRG used as global variable throughout algorithms
3: $\sigma^{v,a} \leftarrow \langle\rangle$, $\forall (v,a) \in V \times A$
4: **for** $\sigma \in \mathcal{L}$ **do** ▷ All traces in the log
5: **for** $e_t \in \sigma$ **do** ▷ All events in the trace
6: **if** $e_{t-1} \neq \emptyset$ **then** ▷ Skip first event to avoid non-existing e_{t-1}
7: **for** $v \in var(e_t)$ **do**
8: **if** $val_v(e_{t-1}) \neq val_v(e_t)$ **then**
9: $\sigma^{v,a} \leftarrow e_t$ ▷ Store the event in the shift sequence
10: **else**
11: $\sigma^{v,a} \leftarrow e_t$, $\forall v \in var(e_t)$ ▷ Store first value as shift
12: **for** $v \in V$ **do**
13: **for** $a \in A$ **do**
14: **if** $|\{val_v(s) \mid s \in S^{v,a}\}| > 1$ **then** ▷ The variable is discriminative
15: **if** $|\sigma^{v,a}| > \underline{minShift} \cdot |\mathcal{L}|$ **then** ▷ Enough shifts occurred for VAP
16: $O(a) \leftarrow O(a) \cup v$
17: $I(a) \leftarrow I(a) \cup v$
▷ Find all inputs set by other activities occurring before certain shifts
18: **for** $(v,a,n) \in \{(v,a,n) \in V \times A \times \mathbb{N} \mid \sigma^{v,a} \neq \emptyset \wedge n \in [1, \underline{maxShift}]\}$ **do**
19: $T_v \leftarrow \{\sigma \mid s^{v,a}_{\sigma,n} \neq \emptyset\}$ ▷ All shifting traces with at least n shifts
20: $DRG \leftarrow drd = (D \cup d = (v,a,n), E, L, C, T_v)$
21: Find input variables(v,a,n,drd)
22: $DRG \leftarrow$ eliminate duplicates(DRG)
23: **return** DRG

- $E \subseteq D \times D$ *the connection between the VASs,*
- $C : E \to \mathbb{R}$*, the correlation to weigh the connections between elements of* D*,*
- $L : \mathbb{P}(V) \times V \to \mathbb{R}$ *the predictive models, and*
- $T \subseteq \mathcal{L}$ *a set of traces for which the model holds.*

Once all possible VASs are captured, they are used as candidate top-level nodes, in a DRG as illustrated in Algorithm 1 lines 18-20. The corresponding traces, i.e., the ones in which there are $[1, \texttt{maxShift}]$ shifts, are analyzed further.

Every decision variable-activity pair (v,a) for which $v \in O(a)$ that is retained after reading the event log is considered as the top node in a decision model. Next, it is checked what other VAPs constituted an influence over the input

variables of the top activity in Algorithm 2. These inputs were established in the previous staged by populating I and O, i.e., the goal is to connect activities that set a v_2 that has a relationship with variable v through a. To this purpose the correlation is calculated between the values of v_2 and v (binary) and by building a predictive model (n-ary) later (line 16). Correlation is calculated by means of RELIEF, which was introduced for two class problems in [35] and extended for multiclass problems in [36]. RELIEF is a feature selection approach similar to information gain or chi square weighing, but is capable of dealing with both numeric and nominal data and is robust to noise and multicollinearity. Hence, it is more suitable than standard Pearson correlation to weigh the importance of variables within the process data. In line 4 of Algorithm 2 only traces in which (at least) n shifts for (v, a) occur are stored. For all these traces, it is checked whether (v_2, a_2)-pairs for a particular shift occurrence n_2 is happening before the influence of a on v (Algorithm 2, lines 5-9). For every (v_2, a_2, n_2) tuple the values of v_2 are stored and eventually checked for their correlation with v in lines 10-13 given that enough traces are supporting this shift order. In case there exist more than 1 shift in the trace of (v, a), this means that previous shifts might have influenced v through a. Hence, autocorrelations are also part of D. The number of possible shifts can be limited by the `maxShift` parameter, as this depends on the event log. Especially longer traces might contain more intricate relations in which the influence of the reoccurrence of shifts is important to discover even autocorrelations. Here it becomes noticeable how partial order restrictions are introduced to ensure that correlations are following the availability of variable information according to the availability in the traces. In Figure 3, it is illustrated how VAPs are appearing throughout a trace, and how their correlations might behave over different shifts.

If combinations of variables manipulated by activities for a certain shift are withheld in M for particular traces, possible predictive models are generated over the different subsets of traces supporting these v, a, n triples in D. There is a vast amount of possible subsets of traces, especially if `maxShift` is high and plenty if different shifts of the same VAPs are considered and passing `minTraces`,

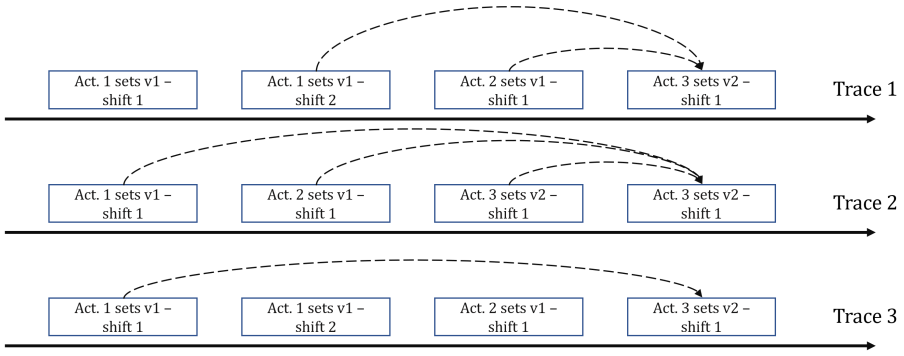

Figure 3: Example of three traces and the existing correlations indicated by dashed lines. These correlations are retrieved and stored to build the predictive models over the traces.

i.e., a subset of the event log with minimum size. To limit discovering all possible combinations of pairs over all traces, a particular rationale is followed to generating possible models which is outlined in Algorithm 3 and is illustrated graphically in Figure 4. Starting from the biggest trace cluster, variables are iteratively added to obtain models that hold for certain parts of the event low. Depending on the overlap of the traces, new clusters are made, that will be tested as new models (i.e. a combination of traces and a set of shifted-based VAPs). By using parameter `minDev`, expressed as a percentage, it is ensured that a variable that is added is sufficiently different and is not only slightly different. To this purpose, the pairs are sorted according to how many traces are withheld during correlation checking. Iteratively, new combinations of variables over a collection of traces are generated, as long as their size is big enough.

Once the different combinations are captured in M, a predictive model is built for all models correlating with (v, a, n) with all other triples on this level in D_2. Various predictive models can be built and many have merits in different situations. As a placeholder, P-MInD uses classification trees and regression trees, however, any other inference algorithms such as artificial neural networks, regressors, and so on can be used instead. The model is stored in L. Before the models are built, categorical variables with a high number of values (>50% of the trace size) are filtered, for they are very unlikely to contain any predictive

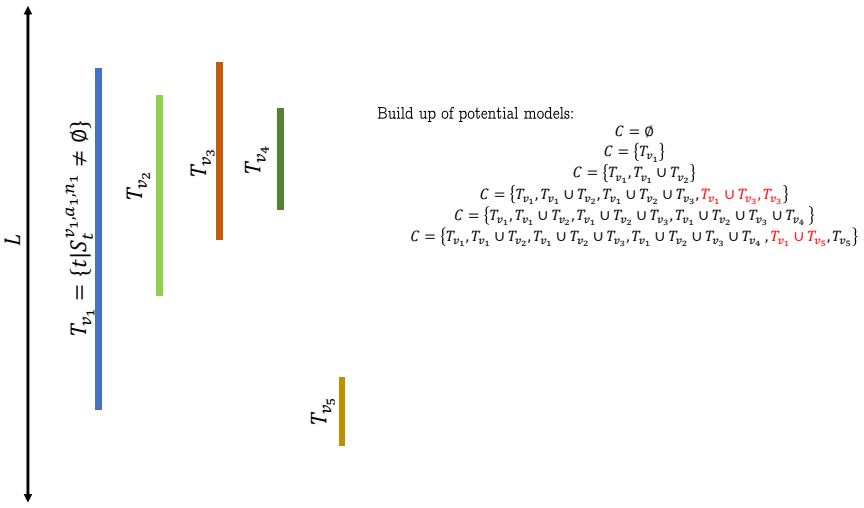

Figure 4: Constructing potential models iteratively starting with the biggest subset of traces. Note that the subsets that are too small ($< |\mathcal{L}| \cdot$ `minTraces`) are indicated in red.

information. All numerical values are normalized first. The same procedure is followed when the RELIEF score is calculated as well. Depending on what trace cluster the set of triples is holding, a new DMN model is established to store that different models hold for different traces. The outcome of the predictive model is measured in Mean Root Squared Error (MRSE) for it can capture the performance of both continuous and discrete values. The latter are captured with a quadratic loss function to be able to calculate the MRSE. The MRSE is a values between 0 and 1, as the continuous values are normalized, and the quadratic loss function ensures an outcome in the same range. It has to exceed the `minSupport` parameter to establish an interesting enough relationship between variables and . Finally, after finding all inputs and the models influencing v, a, n, the algorithm continues recursively to repeat the same procedure for all the triples in D_2 that serve as an input in the particular traces of T.

Finally, P-MInD stops by eliminating duplicates at the end of Algorithm 1. The output consists of all generated DRG tuples. At every stage, it is also checked whether the correlation for a certain pair of VAS has not already been

correlated over a set of the same traces, and whether the combination of d and D_2 in Algorithm 2 over T has not already been used to build a predictive model to ensure no double calculations are performed.

Algorithm 2 Finding all input variables

procedure FIND INPUT VARIABLES($d = (v, a, n), drd = (D, E, L, C, T_v)$)
▷ Construct model with a a decision activity of v as top node
  $M, D_2 \leftarrow \emptyset$
▷ Loop all activities that set variables that serve as an input for v:
  $AV \leftarrow \{(v_2, a_2) \in V \times A \mid v_2 \in I(a) \wedge v_2 \in \{v \mid v \in O(a_2)\}\}$
  for $(v_2, a_2) \in AV$ **do**
    for $d_2 = (v_2, a_2, n_2) \in \{(v_2, a_2, n_2) \in V \times A \times \mathbb{N} \mid \sigma^{v_2, a_2} \neq \emptyset \wedge n_2 \in [1, \underline{maxShift}]\}$ **do**
      $val_v, val_{v_2}, T \leftarrow \emptyset$ ▷ Stores the values of the variables, traces
      for $\sigma \in T_v$ **do**
▷ Include values where shift of a_2 for v_2 occurs before a shifting v in σ:
        if $s^{v_2,a_2}_{\sigma,n_2} < s^{v,a}_{\sigma,n}$ **then**
          $val_v \leftarrow val_v \cup val_v(s^{v,a}_{\sigma,n}), val_{v_2} \leftarrow val_{v_2} \cup val_{v_2}(s^{v_2,a_2}_{\sigma,n_2}), T \leftarrow T \cup \sigma$
▷ Store the inputs that are correlated with values of v over enough traces:
      if $|T| > |\mathcal{L}| \cdot \underline{minTraces} \wedge corr(val_v, val_{v_2}) > \underline{minCorr}$ **then**
        $D_2 \leftarrow d_2$
        $M \leftarrow (d_2, T)$
        $C(d, d_2) \leftarrow corr(val_v, val_{v_2})$ ▷ Correlation between both nodes
  $sort(M)$ ▷ Sort shifts according to $|T|$
▷ Going through different models holding over different trace clusters T:
  for $(D_2, T) \in$ Find input models(M) **do**
▷ D_2 provides the independent variables, v the dependent variable
    $L(D_2, v) \leftarrow$ Build predictive model(D_2, v, T)
    if $L(D_2, v) > \underline{minSupport}$ **then**
      for $d_2 \in D_2$ **do**
        $D \leftarrow d_2$
        $E \leftarrow (d_2, d)$
        if $T = T_v$ **then**
          Find input variables($o_{d_2}, a_{d_2}, n_{d_2}, drd$)
        else ▷ Other trace cluster requires different model to be built
          re-estimate C for T and adjust E accordingly to obtain C_n, E_n
          $DRG \leftarrow drd_n = (D, E_n, L, C_n, T)$
          Find input variables($o_{d_2}, a_{d_2}, n_{d_2}, drd_n$)
  return (D, E)

Algorithm 3 Finding all overlapping models

procedure FIND INPUT MODELS(M) ▷ M a set of tuples of act.-var. pairs over traces
  $models \leftarrow \emptyset$ ▷ Generated models
  $cov \leftarrow \emptyset$ ▷ Covered traces
  for $((v, a, n), T) \in M$ **do**
    for $(D, T_m) \in models$ **do** ▷ Check all possible input models
      if $T = T_m$ **then** ▷ If traces are the same, add to model
        $D \leftarrow D \cup (v, a, n)$
      else if $T \subset T_m \wedge |T| > |\mathcal{L}| \cdot \underline{minTraces}$ **then**
        $models \leftarrow (D \cup (v, a, n), T)$
      else if $T \cup T_m \neq \emptyset \wedge \frac{|T \cup T_m|}{|T|} > \underline{minDev}$ **then**
        $models \leftarrow (D \cup (v, a, n), T \setminus T_m)$
        $cov \leftarrow T$
    if $T \subsetneq cov \wedge |T| > |\mathcal{L}| \cdot \underline{minTraces}$ **then**
      $models \leftarrow ((v, a, n), T)$
      $cov \leftarrow T$
  return $models$

4.3. Consolidation of models and behavioural information

The output consists of a set of DRGs. They can be outputted according to several abstractions. First of all, models can be retrieved and visualized according to the traces to which they apply. I.e., models that are generated for the same traces have the same T. This has the benefit of being able to visualize the decision models over the same behavioural model (see infra). Secondly, it is possible to visualize the models depending on the top-level nodes in the decision models, and bundle all models according to whether they are a more specific, or altered version of the DRG of this top node to capture the difference between trace clusters regarding a certain top-level VAP.

Besides decision model information, it is also possible to mine the traces T for each model to obtain a behavioural model such as BPMN models or Petri nets. This allows to illustrate how the activities and decisions are intertwined and provides a full, holistic overview that captures both perspectives of the recorded execution.

5. DMN model conformance checking

DMN models and DRGs that are discovered from an event log can be useful for many reasons including visualization and description, but also for verification purposes. Conformance checking is deeply rooted in process mining to quantify the quality of the discovered models, but also to measure conformance with other, unseen data.

This section will focus on the conformance along two dimensions. First of all, DRGs have to be corresponding with the underlying process model to ensure that the execution of the process supports the order of decisions. Secondly, the DRGs can be compared to other decision models that were inferred from other event logs.

5.1. Conformance with process model

The DRGs that are part of the final DMN output contain partial orders that are compatible with the process models generated for the trace clusters they are in. However, if other DRGs are verified over a process model, or the the output DRGs of P-MInD over another process model, a conformance check needs to be performed.

To obtain such a check, P-MInD queries the reachability graph [2, 37] of a Petri net mined over the traces in the cluster. The reachability graph of a bounded Petri net is a transition system constructed as follows. The initial marking is the initial state. Every reachable marking from M_0 is a state. Transitions between pairs of states represent the transitions that lead from a marking to another by means of a firing. A state in which no transitions are enabled anymore is called a final state. After its construction, the reachability graph can be queried to verify the sequences in which the activities are used in the DRGs and indicate which partial order relationships in the DRG are not allowed by the behavioral model underpinning its execution. Also reoccurring activities, i.e., activities with multiple shifts in the same model, can be checked for whether they are occurring in the same order as the DRG dictates.

Note that this requires the use of a concurrent discovery model that delivers models that provide the correct class of Petri nets for which the reachability can be calculated. Hence, in the examples, and the implementation in ProM discussed in Section 6, Inductive Miner [38] is used.

P-MInD is a decision-first approach. While it is aligned with the behavioral information in the underlying event log, and hence with discovered process models, this step is rather aimed at positioning the decision models within the global model rather than fixing any misalignments.

5.2. Conformance with other decision models

Decision models contain both the structure of how the decisions are made by the activities, as well as the knowledge nodes or decision logic that is present in the form of predictive models.

To verify the correspondence of a DRG with unseen event logs, the decisions in the form of activity-attribute pairs need to be retrieved as well. Hence, to verify DRGs over event logs, those event logs are mined with P-MInD as well, in order to obtain DRGs that can be matched with the DRGs in the model to be verified. Next, DRGs are compared in order to find whether the decision behavior in the unseen process data corresponds with the decision behavior modeled in the DRGs. In order to achieve this comparison, the coverage between two DRGs $DRG_1 = (D_1, E_1, C_1, L_1, T_1)$ and $DRG_2 = (D_2, E_2, C_2, L_2, T_2)$ is introduced as follows:

$$\begin{aligned} match(DRG_1, DRG_2) = \Sigma_{d \in D_1}(d \in D_2) \\ + \Sigma_{(d_1,d_2) \in E_1}((d_1, d_2) \in E_2) \times C_1(e = (d_1, d_2)) \end{aligned} \tag{1}$$

$$coverage(DRG_1, DRG_2) = \frac{match(DRG_1, DRG_2)}{|D_1| + |E_1|} \tag{2}$$

Equations 1 and 2 allow find the best matching DRG_2 generated from the unseen process data for every DRG_1 in the DRGs contained in the decision model to be verified. In case DRG_1 contains decision logic, traces that are corresponding with DRG_2 can be validated over the predictive models in L_2

in order to obtain the same quality metrics that are used to constructs the models earlier (i.e. AUC or accuracy). Note that other traces can be replayed over DRG_1 as well, however, a high, ideally complete coverage where C_1 is not considered, will ensure that the variables (i.e. an activity-attribute pair for a certain shift) can actually be validated over the predictive model without a high number of missing variables or values.

By combining both checks on the structure of decisions, as well as the decision logic captured in the original model, a comprehensive overview of the quality of conformance can be calculated that pinpoints mismatches, i.e., between DRGs structure in terms of missing activities and arrows, as well as incompatibility in terms of data. E.g., an event log might have the same decision structure, however, the data used for making those decisions might be vastly different from what was used to model the original model for which the verification was performed.

6. Experimental evaluation

In this section, the implementation of P-MInD is illustrated on a real-life event log. To this purpose, the 2017 BPI Challenge log[1] was chosen because it is one of the few commonly used event logs containing a significant activity data component. Next, the influence of the parameters is illustrated on the same log. All high-resolution screenshots can be found online[2].

6.1. Implementation

P-MInD is available as a plugin in the Process Mining (ProM)[3] framework and is available in the ProM repository. The underlying predictive models are created with Weka [4]. It offers all parameters, i.e., minShift, minCorr, minDev, and so on, as well as the possibility to either use parametric or non-parametric

[1] `https://data.4tu.nl/repository/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b`
[2] `https://github.com/JohannesDeSmedt/PMInD`
[3] `http://promtools.org`
[4] `https://www.cs.waikato.ac.nz/ml/weka/`

correlation. Screenshots of output produced by both P-MInD and the P-MInD conformance checker can be found in Figures 5, 6, 7, and 8.

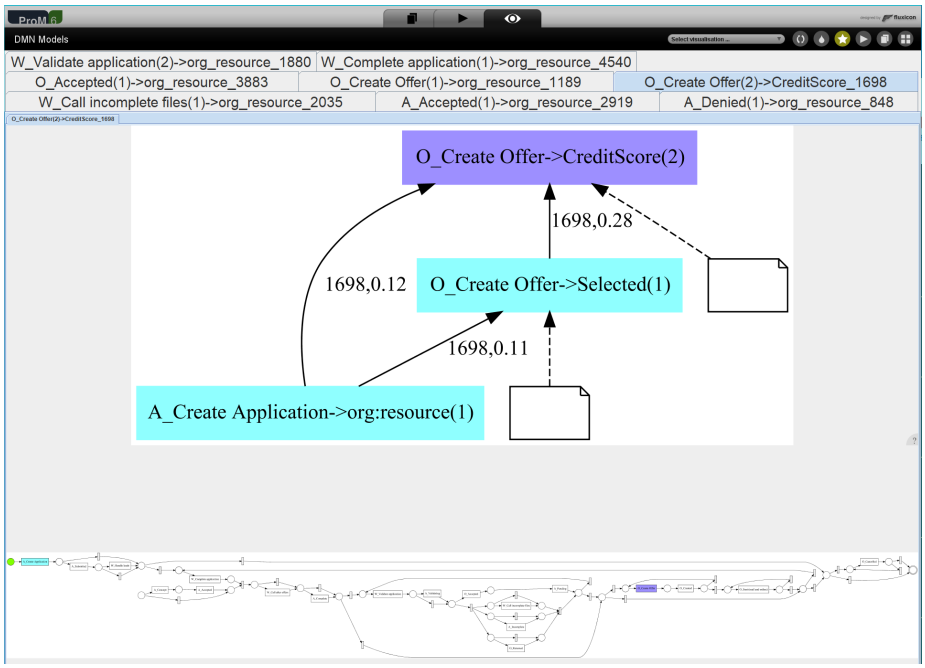

Figure 5: Screenshot of the P-MInD ProM implementation. On the top of the screen, the various top-level decisions (activity(shift)→variable_number of traces) are selectable. Below, the DRG of one of the variants of the decision is displayed, with the corresponding Petri net for the traces supporting the model shown below. Parameters used: (minShift=10, minSupport=60, minTraces=10, minCorr=10, minDev=90)

6.2. 2017 BPI Challenge log

This event log captures a loan application process at a financial institution. It consists of 25,337 events in 1,383 traces over 25 activities, 6 case variables, and 12 activity variables. The set of variables includes the credit score of the applicant (denoted CreditScore), monthly costs of the loan, initial withdrawal amount, the resource handling the activity (denoted org:resource) and so on. The latter are all recorded together with the activities as they occur. Some variables are solely case-based, i.e., application type (denoted ApplicationType),

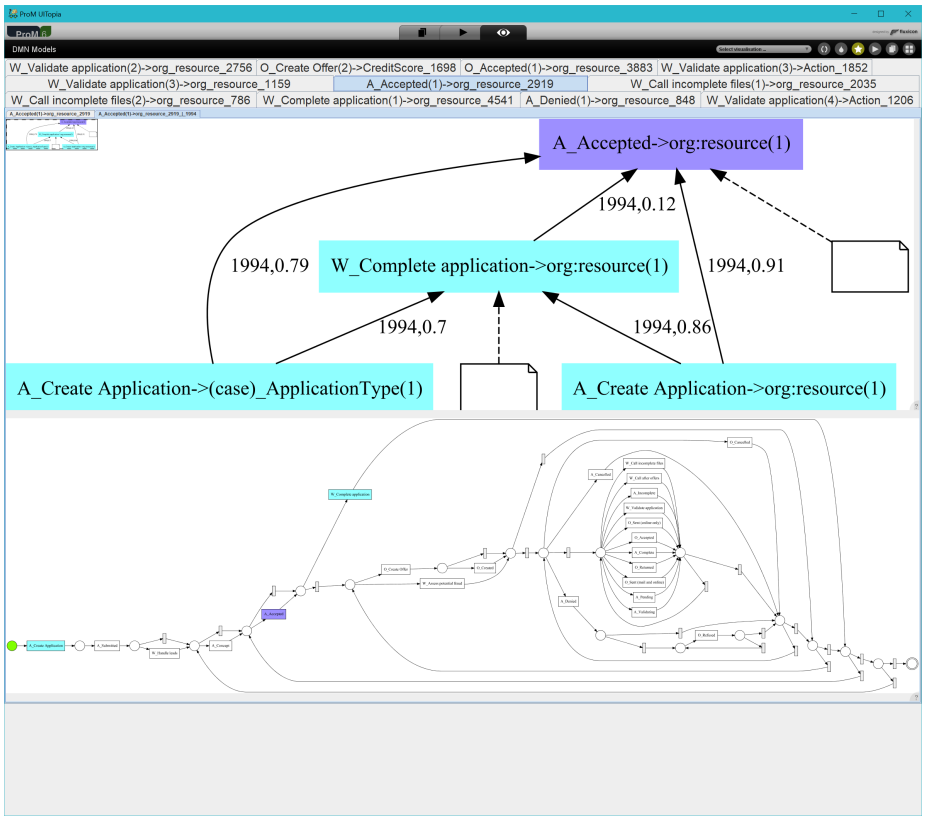

Figure 6: Screenshot of a different decision model from the same run as Figure 5

loan reason, and requested amount. The process follows a relatively straightforward control flow where an application and subsequently a potential offer is created, submitted, handled, accepted or rerouted to the submission state for re-evaluation. The activities are preceded by the letters O, A, and W depending on whether they deal with the activities pertaining to an Offer (e.g. O_Create Offer), the Application (e.g. A_Create Application), or the Work item (e.g. W_Call incomplete files). For an overview, we refer to the various submissions[5].

P-MInD can be used to effectively capture the decisions made throughout. In Figure 5, the output shows how a decision was made regarding the credit

[5] `https://www.win.tue.nl/bpi/doku.php?id=2017:challenge`

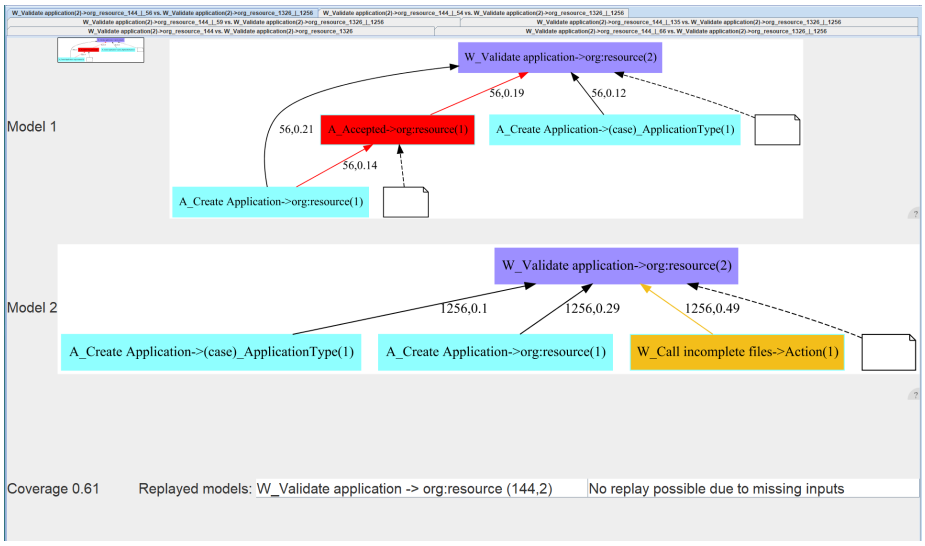

Figure 7: Screenshot of the P-MInD ProM conformance implementation. On the top of the screen, the variants of the top-level decision W_Validate Application for variable org:resource shift 2 are selectable and compared with models from the other DRG set from the other log with the highest coverage. Missing arcs and nodes are indicated in red, missing arcs and nodes missing in the model used for comparison are indicate in a lighter yellowish colour.

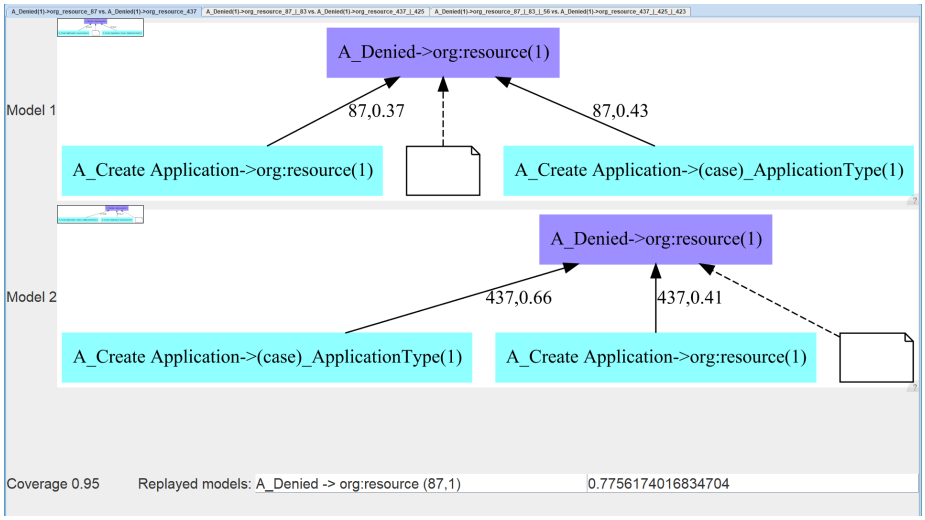

Figure 8: Screenshot of the P-MInD ProM conformance implementation for two conforming DRGs with a high coverage score and replay of traces in event log 2 over the model created with log 1 (upper model).

score after having decided on the credit score previously (as the second shift is being decided on as indicated as (2)). Both the resource, i.e., the person handling the claim, as well as whether the offer was selected previously when the first offer was made (i.e. the execution of O_Create Offer influencing variable Selected to form VAP O_Create Offer→Selected for the first shift (1)) are inputs to the decision on the credit score, the second time it was visited by VAP O_Create Offer→CreditScore(2). The arrows indicate the number of executions for which these relations exists (in this case 1,698), as well as the RELIEF score of the variables that are obtained from correlating org:resource and Selected with CreditScore. The file symbols include the decision model, i.e., predictive model that uses the input variables in order to predict the outcome of the credit score. Note that this only pertains to the subset of the traces in which this decision was taken. Other decisions which pertain to the event log are included on the top of Figure 5 and include decisions regarding what resources (org:resource) are handling the cases, and what variables influence these decisions. Typically, this pertains to certain application types (as illustrated in Figures 7 and 8), and the other resources that handled the case before. They are more prevalent and relevant to more traces in the event log, e.g., up to 4,540 traces for VAP W_Complete application→org_resource(1).

The workflow in Figure 5 (the color scheme indicates what activities are involved) indicates that O_Create Offer is indeed executed multiple times as it contained in different paths that contain loops. The application is rejected, and follows the full workflow to end up at the second offering. This example shows how P-MInD is capable of finding decisions that are not included in the control flow, indeed, there is no influence of the control flow on routing the offer past a certain XOR-split, as O_Create Offer will be executed regardless. It is rather the content of the activity, i.e., its variables, that establish a decision. The importance of using VAPs, especially combined with shifts, is apparent as well. Otherwise, the influence of O_Create Offer over multiple executions during the same case is lost.

In Figure 6, we see what resource (org:resource) is used for A_Accepted, and

how this depends on the case, I.e., the application type. The discovered decision tree for this model, which is outputted by P-MInD separately, is shown in Table 2, and indicates how the application type and the resource determine what user is used to finally make the acceptance decision in A_Accepted.

In Figure 7, two models from different parts of the event log are compared with the conformance checking and how the model also captured autocorrelation (of org:resource), as well as influence of case variables (A_Create Application introduces Application Type at the beginning of every workflow). Notice also how many different activities are present. Although not visualized, the ordering of activities can be visualised as well. A_Create Application sets org:resource first, after which A_Accepted changes it later on in the workflow.

In Figure 9, the conformance checking approach incorporating data approach presented in [14] is shown for the same event log. The Petri net is also mined with Inductive Miner, and is annotated for the data layer by the approach. In hexagonal elements, the variables are all linked to the transitions for which the guards contain the variable. The conditions that allow for a certain event to happen based on the data that is currently holding in the dataset, is added to the arcs. E.g., many arcs indicate which resource is required for executing a particular activity. While the approach provides an in-depth insight into how the variables are set and used for control flow execution, one cannot derive what iteration of a certain loop (of which many are present in the Petri net) is causing a particular activity to be executed. Furthermore, while decision trees are underpinning the creation of the guards, they do not relate activities to other activities directly, only indirectly through the connection with the variables present in the graph. Finally, no case attributes and autocorrelated can be used with this approach. However, in contrast with P-MInD, it is better capable of explaining how the control flow is established and influenced by the data, detailing the exact outline of the whole trace, rather than the separate decisions. Nevertheless, the often spaghetti-like nature of process graphs makes it hard to untangle how data is actually used. Most notably, this process contains many invisible/silent transitions for introducing skipping and looping activities,

A_Create Application → org:resource (1994,1) = User_10: User_33
A_Create Application → org:resource (1994,1) = User_31
— A_Create Application → (case)_ApplicationType (1994,1) = New credit: User_51
— A_Create Application → (case)_ApplicationType (1994,1) = Limit raise: User_18
A_Create Application → org:resource (1994,1) = User_30: User_138)
A_Create Application → org:resource (1994,1) = User_73: User_41
A_Create Application → org:resource (1994,1) = User_14: User_39
A_Create Application → org:resource (1994,1) = User_36: User_52
A_Create Application → org:resource (1994,1) = User_35: User_24
A_Create Application → org:resource (1994,1) = User_13: User_28
A_Create Application → org:resource (1994,1) = User_34: User_97
A_Create Application → org:resource (1994,1) = User_11: User_97
A_Create Application → org:resource (1994,1) = User_18: User_47
A_Create Application → org:resource (1994,1) = User_16: User_53
A_Create Application → org:resource (1994,1) = User_38: User_73
A_Create Application → org:resource (1994,1) = User_37: User_38
A_Create Application → org:resource (1994,1) = User_15: User_38
A_Create Application → org:resource (1994,1) = User_1: User_37
A_Create Application → org:resource (1994,1) = User_2: User_60)
A_Create Application → org:resource (1994,1) = User_3
— A_Create Application → (case)_ApplicationType (1994,1) = New credit: User_10
— A_Create Application → (case)_ApplicationType (1994,1) = Limit raise: User_46
A_Create Application → org:resource (1994,1) = User_109: User_29
A_Create Application → org:resource (1994,1) = User_43: User_97
A_Create Application → org:resource (1994,1) = User_20: User_41
A_Create Application → org:resource (1994,1) = User_25: User_23
A_Create Application → org:resource (1994,1) = User_46: User_28
A_Create Application → org:resource (1994,1) = User_24: User_51
A_Create Application → org:resource (1994,1) = User_23: User_18
A_Create Application → org:resource (1994,1) = User_22: User_15
A_Create Application → org:resource (1994,1) = User_28: User_18
A_Create Application → org:resource (1994,1) = User_49: User_61
A_Create Application → org:resource (1994,1) = User_26: User_3
A_Create Application → org:resource (1994,1) = User_4: User_19
A_Create Application → org:resource (1994,1) = User_5: User_97
A_Create Application → org:resource (1994,1) = User_7: User_42
A_Create Application → org:resource (1994,1) = User_9: User_33

Table 2: Decision table for W_Complete Application→org:resource (1) in Figure 6.

which makes it hard to trace variables and their values back to a particular activity or even execution path. Overall, we recognize P-MInD and the approach complementary where P-MInD gives a more detailed insight traces in which decisions are causing significant correlations, while this approach is addressing the global control flow that holds for all traces. It is worth pointing out how, while both techniques build predictive models, the dependent variables used are different. P-MInD uses the values of the variables in the activities as dependent variables to establish relationships with the other variables present, while the conformance checking approach uses the activity labels as the dependent variable to predict the control flow. This is apparent when comparing the decision rules/trees that are included in Table 2 for P-MInD, and in Figure 9 for the conformance checking approach. As illustrated in Section 2, the technique presented in [6] cannot deal with the control flow behaviour contained in the event log, since loops are present. Hence, the technique cannot be used for benchmarking, and has not been validated over real-life data logs, but only synthetic examples containing no loops.

6.3. Parameter assessment

P-MInD contains a variety of parameters. We illustrate their impact on two excerpts of the 2017 BPI Challenge log, i.e., log 1 contains February-May (7,529 cases) and log 2 contains July-December (18,560 cases), to establish an intuition on their impact on training and to guide users of the algorithm. Results on the average number of edges and nodes, average depth of the DRG and maximal depth, and number of trace clusters per run are contained in Tables 3 and 4.

Clearly, the impact of the minTrace parameter is the most significant, combined with the minCorr parameter. The lower the required correlation for an VAP, the more relations will appear, and the more and larger models will be created. The shift ratio, the initial filter to capture whether there is an influence of an activity over a variable, seems of less importance. If an activity influences a variable, its influence is clear and intense. The minimum support is also of less importance. Only 60% and 70% support are shown, as similar

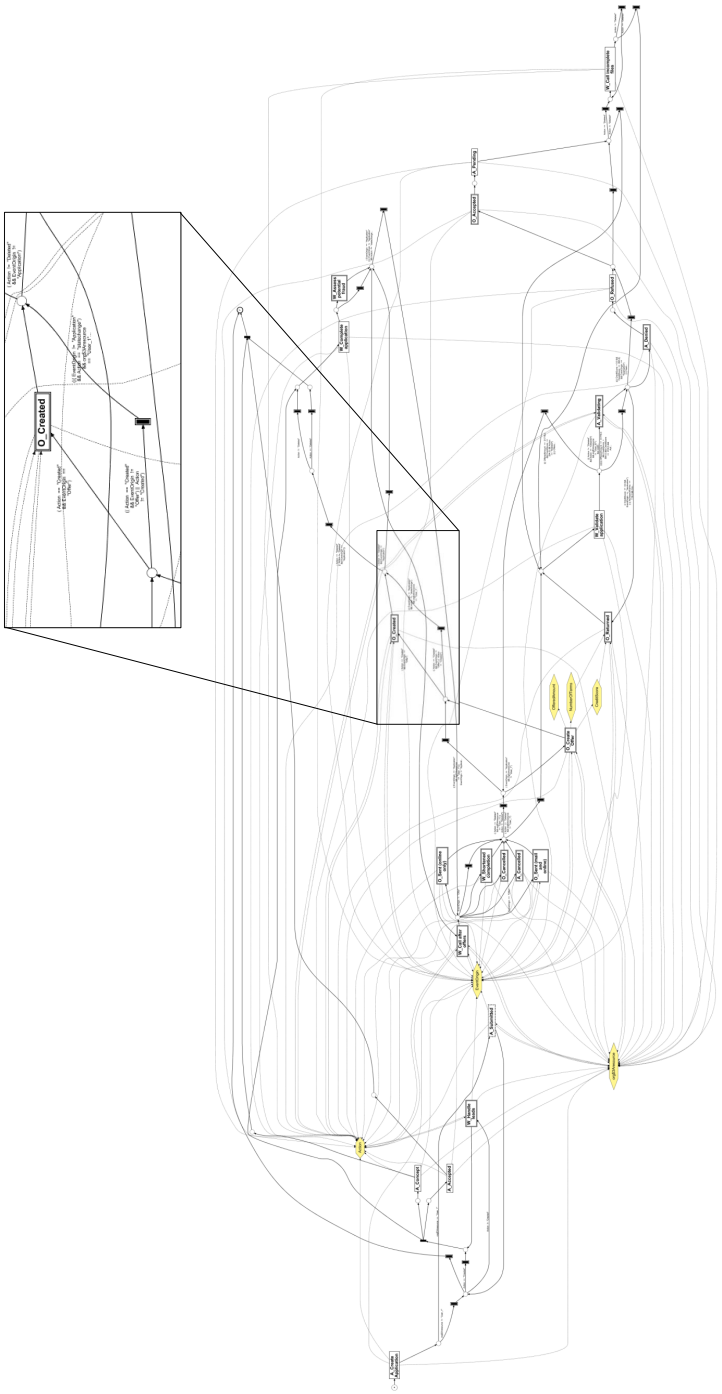

Figure 9: Screenshot of the approach presented in [14] and [39]. It contains the variables in hexagons which are linked up with the transitions that contain guards containing the variables for routing purposes.

results were achieved with lower and higher support values. $\overline{D}$, $\overline{E}$, and $\overline{depth}$ stand for the average number of decisions, the average number of edges, and the average depth of the decision models respectively.

Overall, the average number of nodes per model never exceeds six with no more than 10 relationships between VAPs, meaning that the models are relatively small and comprehensible. The presence of models with a depth of five and visual inspection of the models learns that many models contain autocorrelations. In fact, the latter are very common and can be intuitively linked to the structure of many processes; loops and repetitions are often used to work around the same problem that needs refinement. In this case, the adjustment of the offer details and the revalidation of the offers are causing these repetitions. Finally, it can be noted that there exist more models in the first dataset. Given that it is smaller, it is easier for the algorithm to pick up the relationships. In the bigger log, the thresholds need to be lower to obtain the correlations that exist for only a small amount VAPs, as most of the behaviour in the event logs is straightforward and not causing any convoluted workflows where offers are reconsidered.

6.4. Concluding remarks

We have shown that P-MInD is capable of providing a holistic picture of the data relations in an event log by finding trace clusters in which predictive models relating variables to activities show how decisions are made. E.g., in the case of the 2017 BPI Challenge, insights into the variables influencing the assignment of a credit score (Figure 5) are uncovered by using VAPs for each time the credit score is determined, how this decision is made differently in various trace clusters, and how, in the case of assigning resources (Figure 7), the application type of the case is combined with the insight of which resource created and accepted the case.

By combining all these different insights, i.e., autocorrelation, disentanglement through VAPs with shifts, case attributes, and the independence of - though in consistency with- the workflow, P-MInD brings out holistic decision

minSupport	minShift	minCorr	minTraces	$\overline{D}$	$\overline{E}$	$\overline{\text{depth}}$	max depth	#trace clusters
60	10	10	10	6	9	3	5	72
		10	20	4	5	2	4	28
		10	30	3	3	2	3	9
		20	10	4	5	2	4	32
	10–20–30	20	20	4	3	2	3	18
		20	30	3	2	2	3	6
		30	10	3	2	2	3	9
		30	20	3	2	2	3	7
		30	30	3	2	2	3	3
	20	10	10	6	9	3	5	67
	20–30	10	20	4	5	2	4	28
		10	30	3	3	2	3	9
		20	10	4	5	2	4	30
	30	10	10	6	9	3	5	70
70	10	10	10	6	10	3	5	65
		10	20	4	6	2	4	24
		10	30	3	2	2	3	6
		20	10	4	5	2	4	31
	10–20–30	20	20	4	3	2	3	17
		20	30	3	2	2	3	5
		30	10	3	2	2	3	8
		30	20	3	2	2	3	6
		30	30	3	2	2	3	2
	20	10	10	6	10	3	5	66
	20–30	10	20	4	6	2	4	24
		10	30	3	2	2	3	6
		20	10	4	5	2	4	29
	30	10	10	6	10	3	5	63

Table 3: Models created for the BPI 2017 Challenge February-May. All parameters are expressed in percentages.

models that go beyond a single model, or multiple models trying to span the full event log. Rather, P-MInD addresses coherent clusters of traces enriched with the evolution of the decisions throughout them in the form of autocorrelations with multiple roles for activities depending on their influence over variables. The underlying decision models in the form of decision trees can be used to uncover the current decisions in a process, and be used to alter current business practices. E.g., if certain resources are involved with assigning credit to cases they are not responsible for, this can be uncovered by investigating how the underlying cases in the traces allowed this regardless.

P-MInD differs significantly from other related techniques, mainly in the variables used, and the underlying predictive models that are generated from

minSupport	minShift	minCorr	minTraces	$\overline{D}$	$\overline{E}$	$\overline{\text{depth}}$	max depth	#trace clusters
60	10	10	10	4	5	2	4	38
		10	20	4	5	2	4	25
		10	30	3	3	2	3	13
		20	10	3	2	2	3	9
	10–20–30	20	20	3	2	2	2	5
		20	30	3	2	2	2	4
		30	10	0	0	0	0	0
		30	20	0	0	0	0	0
		30	30	0	0	0	0	0
	20	10	10	4	5	2	4	36
	20–30	10	20	4	5	2	4	25
		10	30	3	3	2	3	13
		20	10	3	3	2	3	8
	30	10	10	4	5	2	4	36
70	10	10	10	4	5	2	4	36
		10	20	4	5	2	4	25
		10	30	3	3	2	3	13
		20	10	3	2	2	3	9
	10–20–30	20	20	3	2	2	2	5
		20	30	3	2	2	2	4
		30	10	0	0	0	0	0
		30	20	0	0	0	0	0
		30	30	0	0	0	0	0
	20	10	10	5	5	2	4	34
	20–30	10	20	4	5	2	4	25
		10	30	3	3	2	3	13
		20	10	3	3	2	3	8
	30	10	10	4	5	2	4	34

Table 4: Models created for the BPI 2017 Challenge July-December. All parameters are expressed in percentages.

them. With P-MInD activities can take part in multiple decisions, as illustrated in the example in Section 2 and in the case study discussed earlier. Furthermore, our approach allows for loops and multiple instantiations of decisions, while providing decision models that are not localised. Rather, the obtained decision models span accross the whole trace execution span in which the control flow and data flow are fully aligned.

Nevertheless, for a view on the control flow, P-MInD can still benefit from being underpinned by approaches such as [39], who tie decisions to control flow elements. Also, a high number of trace clusters might require different users of the process to adequately be informed about the results, and the relation between the clusters.

7. Conclusion

In this paper, we introduced the Process Mining Integrating Decisions framework for the discovery of decision models from event logs. It is the first approach to retrieve holistic decision models from process execution data by making use of variable-activity pairs that are analysed for autocorrelations, looping information over multiple iterations, long-distance dependencies, and the relationship between case variables and activities. P-MInD also does not incorporate control flow information but maintains consistency with the behavioural information in the event log. This allows for getting a deeper insight into how activities are influencing variables during the execution of a process by building predictive models while maintaining compatibility with the workflow and behavioural models. Furthermore, it was illustrated how these models can be used for conformance checking to compare different outcomes according to their decision structure, as well as their suitability in terms of predictive results.

Future work will focus on integrated conformance checking, i.e., blending the control flow and decision perspective into one comprehensive framework to capture how activities are behaving in either models. It will also be investigated to what extend these models can be used for predicting next-activity-in-sequence by blending existing behavioural techniques with the predictive models established with P-MInD.

References

[1] W. van der Aalst, Data Science in Action, in: Process Mining, Springer, 2016.

[2] T. Murata, Petri nets: Properties, analysis and applications, Proceedings of the IEEE 77 (4) (1989) 541–580.

[3] OMG, Business Process Model and Notation (BPMN) 2.0 (2011).

[4] A. Rozinat, W. M. P. van der Aalst, Decision Mining in ProM, in: Business

Process Management, Lecture Notes in Computer Science, Springer, 2006, pp. 420–425.

[5] OMG, Decision Model and Notation (2015).
URL http://www.omg.org/spec/DMN/1.0/

[6] E. Bazhenova, S. Bülow, M. Weske, Discovering Decision Models from Event Logs, in: BIS, Vol. 255 of Lecture Notes in Business Information Processing, Springer, 2016, pp. 237–251.

[7] K. Batoulis, A. Meyer, E. Bazhenova, G. Decker, M. Weske, Extracting Decision Logic from Process Models, in: Advanced Information Systems Engineering - 27th International Conference, CAiSE 2015, Stockholm, Sweden, June 8-12, 2015, Proceedings, 2015, pp. 349–366.

[8] J. De Smedt, F. Hasic, S. K. L. M. vanden Broucke, J. Vanthienen, Towards a Holistic Discovery of Decisions in Process-Aware Information Systems, in: BPM, Vol. 10445 of Lecture Notes in Computer Science, Springer, 2017, pp. 183–199.

[9] J. De Smedt, S. K. L. M. vanden Broucke, J. Obregon, A. Kim, J.-Y. Jung, J. Vanthienen, Decision mining in a broader context: an overview of the current landscape and future directions, in: Business Process Management Workshops, Lecture Notes in Business Information Processing, Springer, 2016.

[10] K. Jensen, L. M. Kristensen, L. Wells, Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems, STTT 9 (3-4) (2007) 213–254.

[11] M. de Leoni, F. M. Maggi, W. M. P. van der Aalst, Aligning Event Logs and Declarative Process Models for Conformance Checking, in: BPM, Vol. 7481 of Lecture Notes in Computer Science, Springer, 2012, pp. 82–97.

[12] F. Mannhardt, M. de Leoni, H. A. Reijers, W. M. P. van der Aalst, Decision Mining Revisited - Discovering Overlapping Rules, in: CAiSE, Vol. 9694 of Lecture Notes in Computer Science, Springer, 2016, pp. 377–392.

[13] M. de Leoni, M. Dumas, L. García-Bañuelos, Discovering Branching Conditions from Business Process Execution Logs, in: FASE, Vol. 7793 of Lecture Notes in Computer Science, Springer, 2013, pp. 114–129.

[14] M. de Leoni, W. M. P. van der Aalst, Data-aware process mining: discovering decisions in processes using alignments, in: Proceedings of the 28th annual ACM symposium on applied computing, ACM, 2013, pp. 1454–1461.

[15] M. de Leoni, F. Mannhardt, Decision discovery in business processes.

[16] G. T. Lakshmanan, D. Shamsi, Y. N. Doganata, M. Unuvar, R. Khalaf, A markov prediction model for data-driven semi-structured business processes, Knowl. Inf. Syst. 42 (1) (2015) 97–126.

[17] E. Bazhenova, M. Weske, Deriving Decision Models from Process Models by Enhanced Decision Mining, in: Business Process Management Workshops, Vol. 256 of Lecture Notes in Business Information Processing, Springer, 2015, pp. 444–457.

[18] K. Batoulis, A. Baumgraß, N. Herzberg, M. Weske, Enabling dynamic decision making in business processes with DMN, in: Business Process Management Workshops, Vol. 256 of Lecture Notes in Business Information Processing, Springer, 2015, pp. 418–431.

[19] T. Savickas, O. Vasilecas, Belief network discovery from event logs for business process analysis, Computers in Industry 100 (2018) 258–266.

[20] F. Mannhardt, M. de Leoni, H. A. Reijers, W. M. P. van der Aalst, Data-driven process discovery: revealing conditional infrequent behavior from event logs, in: Advanced Information Systems Engineering (CAiSE), Springer, 2017.

[21] M. de Leoni, P. Felli, M. Montali, A holistic approach for soundness verification of decision-aware process models, in: ER, Vol. 11157 of Lecture Notes in Computer Science, Springer, 2018, pp. 219–235.

[22] M. de Leoni, W. M. P. van der Aalst, M. Dees, A General Framework for Correlating Business Process Characteristics, in: BPM, Vol. 8659 of Lecture Notes in Computer Science, Springer, 2014, pp. 250–266.

[23] A. Kim, J. Obregon, J.-Y. Jung, Constructing Decision Trees from Process Logs for Performer Recommendation, in: Business Process Management Workshops, Vol. 171 of Lecture Notes in Business Information Processing, Springer, 2013, pp. 224–236.

[24] I. T. P. Vanderfeesten, H. A. Reijers, W. M. P. van der Aalst, Product Based Workflow Support: Dynamic Workflow Execution, in: CAiSE, Vol. 5074 of Lecture Notes in Computer Science, Springer, 2008, pp. 571–574.

[25] H. van der Aa, H. Leopold, K. Batoulis, M. Weske, H. A. Reijers, Integrated Process and Decision Modeling for Data-Driven Processes, in: Business Process Management Workshops, Vol. 256 of Lecture Notes in Business Information Processing, Springer, 2015, pp. 405–417.

[26] R. Hull, E. Damaggio, F. Fournier, M. Gupta, F. F. T. H. III, S. Hobson, M. H. Linehan, S. Maradugu, A. Nigam, P. N. Sukaviriya, R. Vaculín, Introducing the Guard-Stage-Milestone Approach for Specifying Business Entity Lifecycles, in: WS-FM, Vol. 6551 of Lecture Notes in Computer Science, Springer, 2010, pp. 1–24.

[27] V. Popova, D. Fahland, M. Dumas, Artifact Lifecycle Discovery, Int. J. Cooperative Inf. Syst. 24 (1).

[28] G. Li, R. M. de Carvalho, W. M. P. van der Aalst, Automatic discovery of object-centric behavioral constraint models, in: BIS, Vol. 288 of Lecture Notes in Business Information Processing, Springer, 2017, pp. 43–58.

[29] D. Calvanese, M. Dumas, Ü. Laurson, F. M. Maggi, M. Montali, I. Teinemaa, Semantics and analysis of DMN decision tables, in: BPM, Vol. 9850 of Lecture Notes in Computer Science, Springer, 2016, pp. 217–233.

[30] D. Calvanese, M. Dumas, F. M. Maggi, M. Montali, Semantic DMN: formalizing decision models with domain knowledge, in: RuleML+RR, Vol. 10364 of Lecture Notes in Computer Science, Springer, 2017, pp. 70–86.

[31] T. Biard, A. L. Mauff, M. Bigand, J. P. Bourey, Separation of decision modeling from business process modeling using new "decision model and notation" (DMN) for automating operational decision-making, in: PRO-VE, Vol. 463 of IFIP Advances in Information and Communication Technology, Springer, 2015, pp. 489–496.

[32] F. Hasic, J. D. Smedt, J. Vanthienen, Augmenting processes with decision intelligence: Principles for integrated modelling, Decision Support Systems 107 (2018) 1–12.

[33] F. Hasić, J. De Smedt, J. Vanthienen, A Service-Oriented Architecture Design of Decision-Aware Information Systems: Decision as a Service, in: On the Move to Meaningful Internet Systems. OTM 2017 Conferences, Springer International Publishing, Cham, 2017, pp. 353–361.

[34] F. E. A. Horita, J. P. de Albuquerque, V. Marchezini, E. M. Mendiondo, Bridging the gap between decision-making and emerging big data sources: An application of a model-based framework to disaster management in brazil, Decision Support Systems 97 (2017) 12–22.

[35] K. Kira, L. A. Rendell, A practical approach to feature selection, in: ML, Morgan Kaufmann, 1992, pp. 249–256.

[36] I. Kononenko, Estimating attributes: Analysis and extensions of RELIEF, in: ECML, Vol. 784 of Lecture Notes in Computer Science, Springer, 1994, pp. 171–182.

[37] W. Reisig, G. Rozenberg, Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, the volumes are based on the Advanced Course on Petri Nets, held in Dagstuhl, September 1996, Vol. 1491 of Lecture Notes in Computer Science, Springer, 1998.

[38] S. J. J. Leemans, D. Fahland, W. M. P. van der Aalst, Discovering Block-Structured Process Models from Event Logs - A Constructive Approach, in: Petri Nets, Vol. 7927 of Lecture Notes in Computer Science, Springer, 2013, pp. 311–329.

[39] F. Mannhardt, M. de Leoni, H. A. Reijers, W. M. P. van der Aalst, Balanced multi-perspective checking of process conformance, Computing 98 (4) (2016) 407–437.